Python for AI Automation — Cheat Sheet

aibx.io · 2026 Edition

Basics · APIs · AI Integrations

VARIABLES & DATA TYPES

```
name = "Ara"          # string
age  = 10             # integer
rate = 4.9            # float
active = True         # boolean
```

LISTS

```
tools = ["Claude", "GPT", "Gemini"]
tools[0]              # Claude
tools.append('Llama') # add item
tools.remove('GPT')   # remove item
len(tools)            # length: 3
```

DICTIONARIES

```
user = {
  "name": "Ara",
  "role": "AI Engineer"
}
user["name"]          # Ara
user["role"] = "Lead" # update
```

LOOPS

```
for tool in tools:
    print(tool)

for i in range(1, 5): # 1,2,3,4
    print(i)

i = 0
while i < 3:
    i += 1
```

CONDITIONALS

```
if age > 18:
    print("Adult")
elif age == 18:
    print("Just turned!")
else:
    print("Minor")
```

FUNCTIONS

```
def greet(name):
    return f"Hello, {name}"

# Call it
result = greet("AIBX")
print(result)  # Hello, AIBX

# Default argument
def greet(name="World"):
    return f"Hello, {name}"
```

FILE HANDLING

```
# Write a file
with open("notes.txt", "w") as f:
    f.write("AIBX Guide")

# Read a file
with open("notes.txt", "r") as f:
    content = f.read()
    print(content)
```

ERROR HANDLING

```
try:
    result = int("abc")
except ValueError as e:
    print(f"Error: {e}")
except Exception as e:
    print('Unexpected:', e)
finally:
    print("Always runs")
```

HTTP API REQUEST

```
import requests

url = 'https://api.github.com'
response = requests.get(url)

# Check status
print(response.status_code) # 200

# Parse JSON response
data = response.json()
```

CALLING AI MODELS (OPENAI)

```
from openai import OpenAI

client = OpenAI(api_key='YOUR_KEY')

res = client.chat.completions.create(
    model="gpt-4.1",
    messages=[{
        "role": "user",
        "content": "Explain agents"
    }]
)
print(res.choices[0].message.content)
```

CALLING AI MODELS (ANTHROPIC)

```python
import anthropic

client = anthropic.Anthropic(
    api_key='YOUR_KEY'
)

msg = client.messages.create(
    model="claude-sonnet-4-5",
    max_tokens=1000,
    messages=[{
        "role": "user",
        "content": "Hello!"
    }]
)
print(msg.content[0].text)
```

LIST COMPREHENSIONS

```python
# Standard loop → comprehension
squares = [x**2 for x in range(5)]
# [0, 1, 4, 9, 16]

# With condition
evens = [x for x in range(10)
         if x % 2 == 0]

# Dict comprehension
scores = {k: v*2 for k,v in d.items()}
```

IMPORTING LIBRARIES

```python
import os
import json
import requests
from pathlib import Path
from datetime import datetime

# Install a library
# pip install requests openai anthropic
```

WORKING WITH JSON

```python
import json

# Parse JSON string → dict
data = json.loads('{"name":"Ara"}')

# Dict → JSON string
text = json.dumps(data, indent=2)

# Read / write JSON file
with open("data.json","r") as f:
    config = json.load(f)
```

ESSENTIAL LIBRARIES

requests	HTTP calls & REST APIs
openai	OpenAI / GPT models
anthropic	Claude AI models
pandas	Data analysis & CSV
FastAPI	Build REST backends
flask	Lightweight web apps
asyncio	Async automation
pathlib	File & path management
rich	Pretty terminal output
langchain	LLM pipelines & agents

4-STAGE LEARNING PATH

Stage 1 — Basics

Variables, loops, functions, lists, dicts

Stage 2 — Automation

Files, APIs, JSON, scheduling

Stage 3 — AI Integrations

OpenAI/Claude SDKs, prompts, embeddings

Stage 4 — Production

FastAPI, Docker, async, deployment

BEGINNER PROJECT IDEAS

- **AI Research Assistant**
  Topic → AI API → summarize → export
- **Folder Automation**
  Watch folder → process files → notify
- **AI Content Pipeline**
  Generate drafts → format → publish
- **API Monitor Dashboard**
  Track uptime, latency & failures

RESOURCES

→ docs.python.org/3
→ docs.anthropic.com
→ platform.openai.com/docs
→ python.langchain.com/docs
→ fastapi.tiangolo.com
→ realpython.com
→ automatetheboringstuff.com